

MaTrust: An Effective Multi-Aspect Trust Inference Model

Yuan Yao
State Key Laboratory for Novel
Software Technology, China
yyao@smail.nju.edu.cn

Hanghang Tong
IBM T.J. Watson Research
htong@us.ibm.com

Xifeng Yan
University of California at
Santa Barbara, USA
xyan@cs.ucsb.edu

Feng Xu
State Key Laboratory for Novel
Software Technology, China
xf@nju.edu.cn

Jian Lu
State Key Laboratory for Novel
Software Technology, China
lj@nju.edu.cn

ABSTRACT

Trust is a fundamental concept in many real-world applications such as e-commerce and peer-to-peer networks. In these applications, users can generate local opinions about the counterparts based on direct experiences, and these opinions can then be aggregated to build trust among unknown users. The mechanism to build new trust relationships based on existing ones is referred to as *trust inference*. State-of-the-art trust inference approaches employ the *transitivity* property of trust by propagating trust along connected users. In this paper, we propose a novel trust inference model (*MaTrust*) by exploring an equally important property of trust, i.e., the *multi-aspect* property. MaTrust directly characterizes multiple *latent* factors for each trustor and trustee from the locally-generated trust relationships. Furthermore, it can naturally incorporate prior knowledge as *specified* factors. These factors in turn serve as the basis to infer the unseen trustworthiness scores. Experimental evaluations on real data sets show that the proposed MaTrust *significantly* outperforms several benchmark trust inference models in both effectiveness and efficiency.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database applications—*Data mining*

General Terms

Algorithms, Experimentation

Keywords

Trust inference, transitivity property, multi-aspect property, latent factors, prior knowledge

1. INTRODUCTION

Trust is essential to reduce uncertainty and boost collaborations in many real-world applications including e-commerce [10], peer-to-peer networks [11], semantic Web [25], etc. In these applications, trust inference is widely used as the mechanism to build trust

among unknown users. Typically, trust inference takes as input the existing trust ratings locally generated through direct interactions, and outputs an estimated trustworthiness score from a trustor to an unknown trustee. This trustworthiness score indicates to what degree the trustor could expect the trustee to perform a given action.

The basic assumption behind most of the existing trust inference methods is the *transitivity* property of trust [17], which basically means that if Alice trusts Bob and Bob trusts Carol, Alice might also trust Carol to some extent. These methods (see Section 2 for a review), referred to as *trust propagation* models as a whole, have been widely studied and successfully applied in many real-world settings [7, 36, 17, 13, 8, 20].

In addition to transitivity, a few trust inference models explore another equally important property, the *multi-aspect* of trust [5, 27]. The basic assumption behind the multi-aspect methods is that trust is the composition of multiple factors, and different users may have different preferences for these factors. For example, in e-commerce, some users might care more about the factor of delivering time, whereas others give more weight to the factor of product price. However, the existing multi-aspect trust inference methods [26, 34, 30, 28] require as input more information (e.g., the delivering time as well as user's preference for it) in addition to locally-generated trust ratings, and therefore become infeasible in many trust networks where such information is not available.

Another limitation in existing trust inference models is that they tend to ignore some important prior knowledge during the inference procedure. In social science community, it is commonly known that *trust bias* is an integrated part in the final trust decision [29]. Therefore, it would be helpful if we can incorporate such prior knowledge into the trust inference model. In computer science community, researchers begin to realize the importance of trust bias, and a recent work [21] models trustor bias as the propensity of a trustor to trust others.

In this paper, we focus on improving the trust inference accuracy by integrating the multi-aspect property and trust bias together, and the result is the proposed trust inference model *MaTrust*. Different from the existing multi-aspect trust inference methods, MaTrust *directly* characterizes multiple latent factors for each trustor and trustee from existing trust ratings. In addition, MaTrust can naturally incorporate the priori knowledge (e.g., trust bias) as several specified factors. In particular, we consider three types of trust bias, i.e., *global bias*, *trustor bias*, and *trustee bias*. We will refer to the characterized latent and specified factors as *stereotypes*

in the following. Finally, the characterized stereotypes are in turn used to estimate the trust ratings between unknown users. Compared with the existing multi-aspect methods, the proposed method is more general since it does not require any information as input other than the locally-generated trust ratings. Compared with the trust propagation methods, our experimental evaluations on real data sets indicate that the proposed MaTrust is *significantly* better in both effectiveness and efficiency.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the definition of the trust inference problem. Section 4 describes our optimization formulation for the problem defined in the previous section and shows how to incorporate the priori knowledge. Section 5 presents the inference algorithm to solve the formulation. Section 6 provides experimental results. Section 7 concludes the paper.

2. RELATED WORK

In this section, we introduce related trust inference models including trust propagation models, multi-aspect trust inference models, and other related methods.

2.1 Trust Propagation Models

To date, a large body of trust inference models are based on trust propagation where trust is propagated along connected users in the trust network, i.e., the web of locally-generated trust ratings. Based on the interpretation of trust propagation, we further categorize these models into two classes: *path interpretation* and *component interpretation*.

In the first category of path interpretation, trust is propagated along a path from the trustor to the trustee, and the propagated trust from multiple paths can be combined to form a final trustworthiness score. For example, Wang and Singh [31, 32] as well as Hang et al. [8] propose operators to concatenate trust along a path and aggregate trust from multiple paths. Liu et al. [16] argue that not only trust values but social relationships and recommendation role are important for trust inference. In contrast, there is no explicit concept of paths in the second category of component interpretation. Instead, trust is treated as random walks on a graph or on a Markov chain [25]. Examples of this category include [7, 20, 36, 13, 23].

Different from these existing trust propagation models, the proposed MaTrust focuses on the multi-aspect of trust and directly characterizes several factors/aspects from the existing trust ratings. Compared with trust propagation models, our MaTrust has several unique advantages, including (1) multi-aspect property of trust can be captured; and (2) various types of prior knowledge can be naturally incorporated. In addition, one known problem about these propagation models is the slow on-line response speed [35], while MaTrust enjoys the *constant* on-line response time and the *linear* scalability for pre-computation.

2.2 Multi-Aspect Trust Inference Models

Researchers in social science have explored the multi-aspect property of trust for several years [27]. In computer science, there also exist a few trust inference models that *explicitly* explore the multi-aspect property of trust. For example, Xiong and Liu [34] model the value of the transaction in trust inference; Wang and Wu [30] take competence and honesty into consideration; Tang et al. [28] model aspect as a set of products that are similar to each other un-

der product review sites; Sabater and Sierra [26] divide trust in e-commerce environment into three aspects: price, delivering time, and quality.

However, all these existing multi-aspect trust inference methods require more information (e.g., value of transaction as well as user's preference for it, product and its category, etc.) and therefore become infeasible when such information is not available. In contrast, MaTrust does not require any information other than the locally-generated trust ratings, and could therefore be used in more general scenarios.

In terms of trust bias, Mishra et al. [21] propose an iterative algorithm to compute trustor bias. In contrast, our focus is to incorporate various types of trust bias as specified factors/aspects to increase the accuracy of trust inference.

2.3 Other Related Methods

Recently, researchers begin to apply machine learning models for trust inference. Nguyen et al. [22] learn the importance of several trust-related features derived from a social trust framework. Our method takes a further step here by simultaneously learning the latent factors and the importance of bias. Seemingly similar concept of stereotype for trust inference is also used by Liu et al. [18] and Burnett et al. [3]. These methods learn the stereotypes from the user profiles of the trustees that the trustor has interacted with, and then use these stereotypes to reflect the trustor's first impression about unknown trustees. In contrast, MaTrust builds its stereotypes based on the existing trust ratings to capture multiple aspects for trust inference. There are also some recent work on using link prediction approaches to predict the *binary* trust/distrust relationship [14, 4, 9]. In this paper, we focus on the more general case where we want to infer a *continuous* trustworthiness score from the trustor to the trustee.

Finally, multi-aspect methods have been extensively studied in recommender systems [1, 12, 19]. In terms of methodology, the closest related work is the collaborative filtering algorithm in [12], which can be viewed as a special case of the proposed MaTrust. As mentioned before, our MaTrust is more general by learning the optimal weights for the prior knowledge and it leads to further performance improvement. On the application side, the goal of recommender systems is to predict users' flavors of items. It is interesting to point out that (1) on one hand, trust between users could help to predict the flavors as we may give more weight to the recommendations provided by trusted users; (2) on the other hand, trust itself might be affected by the similarity of flavors since users usually trust others with a similar taste [6]. Although out of the scope of this paper, using recommendation to further improve trust inference accuracy might be an interesting topic for future work.

3. PROBLEM DEFINITION

In this section, we formally define our multi-aspect trust inference problem. Table 1 lists the main symbols we use throughout the paper.

Following conventions, we use bold capital letters for matrices, and bold lower case letters for vectors. For example, we use a partially observed matrix \mathbf{T} to model the locally-generated trust relationships, where the existing/observed trust relationships are represented as non-zero trust ratings and non-existing/unobserved relationships are represented as '?'. As for the observed trust rating, we represent it as a real number between 0 and 1 (a higher rating

Table 1: Symbols.

Symbol	Definition and Description
\mathbf{T}	the partially observed trust matrix
\mathbf{F}	the characterized trustor matrix
\mathbf{G}	the characterized trustee matrix
\mathbf{F}_0	the sub-matrix of \mathbf{F} for latent factors
\mathbf{G}_0	the sub-matrix of \mathbf{G} for latent factors
\mathbf{T}'	the transpose of matrix \mathbf{T}
$\mathbf{T}(i, j)$	the element at the i^{th} row and j^{th} column of matrix \mathbf{T}
$\mathbf{T}(i, :)$	the i^{th} row of matrix \mathbf{T}
$\mathbf{T}(i, :)'$	the transpose of vector $\mathbf{T}(i, :)$
\mathcal{K}	the set of observed trustor-trustee pairs in \mathbf{T}
μ	the global bias
\mathbf{x}	the vector of trustor bias
\mathbf{y}	the vector of trustee bias
$\mathbf{x}(i)$	the i^{th} element of vector \mathbf{x}
n	the number of users
c	the number of specified factors
r	the number of latent factors
s	the number of all factors, $s = c + r$
$\alpha_1, \alpha_2, \alpha_3$	the coefficients for specified factors
u	the trustor
v	the trustee
m_1, m_2	the maximum iteration number
ξ_1, ξ_2	the threshold to stop the iteration

means more trustworthiness). We use calligraphic font \mathcal{K} to denote the set of observed trustor-trustee indices in \mathbf{T} . Similar to Matlab, we also denote the i^{th} row of matrix \mathbf{T} as $\mathbf{T}(i, :)$, and the transpose of a matrix with a prime. In addition, we denote the number of users as n and the number of characterized factors as s . Without loss of generality, we assume that the goal of our trust model is to infer the unseen trust relationship from the user u to another user v , where u is the trustor and v is the unknown trustee to u .

Based on these notations, we first define the basic trust inference problem as follows:

PROBLEM 1. The Basic Trust Inference Problem

Given: an $n \times n$ partially observed trust matrix \mathbf{T} , a trustor u , and a trustee v , where $1 \leq u, v \leq n$ ($u \neq v$) and $\mathbf{T}(u, v) = '?'$;

Find: the estimated trustworthiness score $\hat{\mathbf{T}}(u, v)$.

In the above problem definition, given a trustor-trustee pair, the only information we need as input is the locally-generated trust ratings (i.e., the partially observed matrix \mathbf{T}). The goal of trust inference is to infer the new trust ratings (i.e., unseen/unobserved trustworthiness scores in the partially observed matrix \mathbf{T}) by collecting the knowledge from existing trust relationships. In this paper, we assume that we can access such existing trust relationships. For instance, these relationships could be collected by central servers in a centralized environment like eBay, or by individuals in a distributed environment like EigenTrust [11]. How to collect these trust relationships is out of the scope of this work.

In this paper, we propose a multi-aspect model for such trust inference in Problem 1. That is, we want to infer an $n \times s$ trustor

matrix \mathbf{F} whose element indicates to what extent the corresponding person trusts others wrt a specific aspect/factor. Similarly, we want to infer another $n \times s$ trustee matrix \mathbf{G} whose element indicates to what extent the corresponding person is trusted by others wrt a specific aspect/factor. Such trustor and trustee matrices are in turn used to infer the unseen trustworthiness scores. Based on the basic trust inference problem, we define the multi-aspect trust inference problem under MaTrust as follows:

PROBLEM 2. The MaTrust Trust Inference Problem

Given: an $n \times n$ partially observed trust matrix \mathbf{T} , the number of factors s , a trustor u , and a trustee v , where $1 \leq u, v \leq n$ ($u \neq v$) and $\mathbf{T}(u, v) = '?'$;

Find: (1) an $n \times s$ trustor matrix \mathbf{F} and an $n \times s$ trustee matrix \mathbf{G} ; (2) the estimated trustworthiness score $\hat{\mathbf{T}}(u, v)$.

3.1 An Illustrative Example

To further illustrate our MaTrust trust inference problem (Problem 2), we give an intuitive example as shown in Fig. 1.

In this example, we observe several locally-generated pair-wise trust relationships between five users (e.g., 'Alice', 'Bob', 'Carol', 'David', and 'Elva') as shown in Fig. 1(a). Each observation contains a trustor, a trustee, and a numerical trust rating from the trustor to the trustee. We then model these observations as a 5×5 partially observed matrix \mathbf{T} (see Fig. 1(b)) where $\mathbf{T}(i, j)$ is the trust rating from the i^{th} user to the j^{th} user if the rating is observed and $\mathbf{T}(i, j) = '?'$ otherwise. Notice that we do not consider self-ratings and thus represent the diagonal elements of \mathbf{T} as '/'. By setting the number of factors $s = 2$, our goal is to infer two 5×2 matrices \mathbf{F} and \mathbf{G} (see Fig. 1(c)) from the input matrix \mathbf{T} . Each row of the two matrices is the stereotype for the corresponding user, and each column of the matrices represents a certain aspect/factor in trust inference (e.g., 'delivering time', 'product price', etc). For example, we can see that Alice trusts others strongly wrt both 'delivering time' and 'product price' (based on matrix \mathbf{F}), and she is in turn moderately trusted by others wrt these two factors (based on matrix \mathbf{G}). On the other hand, both Bob and Carol put more emphasis on the delivering time, while David and Elva care more about the product price.

Once \mathbf{F} and \mathbf{G} are inferred, we can use these two matrices to estimate the unseen trustworthiness scores (i.e., the '?' elements in \mathbf{T}). For instance, the trustworthiness from Carol to Alice can be estimated as $\hat{\mathbf{T}}(3, 1) = \mathbf{F}(3, :)\mathbf{G}(1, :)' = 0.5$. This estimation is reasonable because Carol has the same stereotype as Bob and the trustworthiness score from Bob to Alice is also 0.5. As another example, David and Elva have similar preferences (i.e., the same stereotypes), and thus we conjecture that they would trust each other (i.e., $\hat{\mathbf{T}}(4, 5) = \mathbf{F}(4, :)\mathbf{G}(5, :)' = 1$). In the rest of the paper, we will mainly focus on how to characterize \mathbf{F} and \mathbf{G} from the partially observed input matrix \mathbf{T} .

4. THE PROPOSED OPTIMIZATION FORMULATION

In this section, we present our optimization formulation for the problem defined in the previous section. We start with the basic form, and then show how to incorporate the trust bias as specified factors followed by its equivalent formulation. Finally, we discuss some generalizations of our formulation.

Trustor	Trustee	Rating
Alice	Bob	1
Alice	Carol	1
Alice	David	1
Alice	Elva	1
Bob	Alice	0.5
Bob	Carol	1
Carol	Bob	1
David	Alice	0.5
Elva	Alice	0.5
Elva	David	1

$$\mathbf{T} = \begin{matrix} & \begin{matrix} \text{Alice} & \text{Bob} & \text{Carol} & \text{David} & \text{Elva} \end{matrix} \\ \begin{matrix} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \\ \text{Elva} \end{matrix} & \begin{bmatrix} / & 1 & 1 & 1 & 1 \\ 0.5 & / & 1 & ? & ? \\ ? & 1 & / & ? & ? \\ 0.5 & ? & ? & / & ? \\ 0.5 & ? & ? & 1 & / \end{bmatrix} \end{matrix} \begin{matrix} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \\ \text{Elva} \end{matrix} \left. \begin{matrix} \\ \\ \\ \\ \end{matrix} \right\} \text{trustors}$$

$$\begin{matrix} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \\ \text{Elva} \end{matrix} \left. \begin{matrix} \\ \\ \\ \\ \end{matrix} \right\} \text{trustees}$$

$$\mathbf{F} = \begin{matrix} \begin{matrix} \text{Delivering time} \\ \text{Product price} \end{matrix} & \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{matrix} \begin{matrix} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \\ \text{Elva} \end{matrix} \left. \begin{matrix} \\ \\ \\ \\ \end{matrix} \right\} \text{trustors}$$

$$\mathbf{G} = \begin{matrix} \begin{matrix} \text{Delivering time} \\ \text{Product price} \end{matrix} & \begin{bmatrix} 0.5 & 0.5 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \end{matrix} \begin{matrix} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \\ \text{Elva} \end{matrix} \left. \begin{matrix} \\ \\ \\ \\ \end{matrix} \right\} \text{trustees}$$

$$\begin{matrix} \text{Alice} \\ \text{Bob} \\ \text{Carol} \\ \text{David} \\ \text{Elva} \end{matrix} \left. \begin{matrix} \\ \\ \\ \\ \end{matrix} \right\} \text{trustees}$$

(a) The observed locally-generated pair-wise trust relationships

(b) The partially observed trust matrix \mathbf{T}

(c) The inferred trustor matrix \mathbf{F} and trustee matrix \mathbf{G}

Figure 1: An illustrative example for MaTrust.

4.1 The Basic Formulation

Formally, Problem 2 can be formulated as the following optimization problem:

$$\min_{\mathbf{F}, \mathbf{G}} \sum_{(i,j) \in \mathcal{K}} (\mathbf{T}(i,j) - \mathbf{F}(i,:) \mathbf{G}(j,:))'^2 + \lambda \|\mathbf{F}\|_{fro}^2 + \lambda \|\mathbf{G}\|_{fro}^2 \quad (1)$$

where λ is a regularization parameter; $\|\mathbf{F}\|_{fro}$ and $\|\mathbf{G}\|_{fro}$ are the Frobenious norm of the trustor and trustee matrices, respectively.

By this formulation, MaTrust aims to minimize the squared error on the set of observed trust ratings. Notice that in Eq. (1), we have two additional regularization terms ($\|\mathbf{F}\|_{fro}^2$ and $\|\mathbf{G}\|_{fro}^2$) to improve the solution stability. The parameter $\lambda \geq 0$ controls the amount of such regularizations. Based on the resulting \mathbf{F} and \mathbf{G} of the above equation, the unseen trustworthiness score $\hat{\mathbf{T}}(u,v)$ can then be estimated by the corresponding stereotypes $\mathbf{F}(u,:)$ and $\mathbf{G}(v,:)$ as:

$$\hat{\mathbf{T}}(u,v) = \mathbf{F}(u,:) \mathbf{G}(v,:)' \quad (2)$$

4.2 Incorporating Bias

The above formulation can naturally incorporate some prior knowledge such as trust bias into the inference procedure. In this paper, we explicitly consider the following three types of trust bias: *global bias*, *trustor bias*, and *trustee bias*, although other types of bias can be incorporated in a similar way.

- The *global bias* represents the average level of trust in the community. The intuition behind this is that users tend to rate optimistically in some reciprocal environments (e.g., e-Commerce [24]) while they are more conservative in others (e.g., security-related applications). As a result, it might be useful to take such global bias into account and we model it as a scalar μ .
- The *trustor bias* is based on the observation that some trustors tend to generously give higher trust ratings than others. This bias reflects the propensity of a given trustor to trust others, and it may vary a lot among different trustors. Accordingly, we can model the trustor bias as vector \mathbf{x} with $\mathbf{x}(i)$ indicating the trust propensity of the i^{th} trustor.
- The third type of bias (*trustee bias*) aims to characterize the fact that some trustees might have relatively higher capability in terms of being trusted than others. Similar to the second type of bias, we model this type of bias as vector \mathbf{y} , where

$\mathbf{y}(j)$ indicates the overall capability of the j^{th} trustee compared to the average.

Each of these three types of bias can be represented as a *specified* factor for our MaTrust model, respectively. By incorporating such bias into Eq. (1), we have the following formulation:

$$\min_{\mathbf{F}, \mathbf{G}} \sum_{(i,j) \in \mathcal{K}} (\mathbf{T}(i,j) - \mathbf{F}(i,:) \mathbf{G}(j,:))'^2 + \lambda \|\mathbf{F}\|_{fro}^2 + \lambda \|\mathbf{G}\|_{fro}^2$$

Subject to:

$$\begin{aligned} \mathbf{F}(:, 1) &= \mu \mathbf{1}, \quad \mathbf{G}(:, 1) = \alpha_1 \mathbf{1} \quad (\text{global bias}) \\ \mathbf{F}(:, 2) &= \mathbf{x}, \quad \mathbf{G}(:, 2) = \alpha_2 \mathbf{1} \quad (\text{trustor bias}) \\ \mathbf{F}(:, 3) &= \alpha_3 \mathbf{1}, \quad \mathbf{G}(:, 3) = \mathbf{y} \quad (\text{trustee bias}) \end{aligned} \quad (3)$$

where α_1, α_2 , and α_3 are the weights of bias that we need to estimate based on the existing trust ratings.

In addition to these three *specified* factors, we refer to the remaining factors in the trustor and trustee matrices as *latent* factors. To this end, we define two $n \times r$ sub-matrices of \mathbf{F} and \mathbf{G} for the latent factors. That is, we define $\mathbf{F}_0 = \mathbf{F}(:, 4 : s)$ and $\mathbf{G}_0 = \mathbf{G}(:, 4 : s)$, where each column of \mathbf{F}_0 and \mathbf{G}_0 corresponds to one latent factor and r is the number of latent factors. With this notation, we have the following equivalent form of Eq. (3):

$$\min_{\mathbf{F}_0, \mathbf{G}_0, \alpha_1, \alpha_2, \alpha_3} \sum_{(i,j) \in \mathcal{K}} (\mathbf{T}(i,j) - (\alpha_1 \mu + \alpha_2 \mathbf{x}(i) + \alpha_3 \mathbf{y}(j) + \mathbf{F}_0(i,:) \mathbf{G}_0(j,:))')^2 + \lambda \|\mathbf{F}\|_{fro}^2 + \lambda \|\mathbf{G}\|_{fro}^2 \quad (4)$$

Notice that there is no coefficient before $\mathbf{F}_0(i,:) \mathbf{G}_0(j,:)$ as it will be automatically absorbed into \mathbf{F}_0 and \mathbf{G}_0 . Once we have inferred all the parameters (i.e., \mathbf{F}_0 , \mathbf{G}_0 , α_1 , α_2 , and α_3) of Eq. (4), the unseen trustworthiness score $\hat{\mathbf{T}}(u,v)$ can be immediately estimated as:

$$\hat{\mathbf{T}}(u,v) = \mathbf{F}_0(u,:) \mathbf{G}_0(v,:)' + \alpha_1 \mu + \alpha_2 \mathbf{x}(u) + \alpha_3 \mathbf{y}(v) \quad (5)$$

In the above formulations, we need to specify the three types of bias, i.e., to compute μ , \mathbf{x} , and \mathbf{y} . Remember that the only information we need for MaTrust is the existing trust ratings. Therefore,

we simply estimate the bias information from \mathbf{T} as follows:

$$\begin{cases} \mu = \sum_{(i,j) \in \mathcal{K}} \mathbf{T}(i,j) / |\mathcal{K}| \\ \mathbf{x}(i) = \sum_{j, (i,j) \in \mathcal{K}} \mathbf{T}(i,j) / |\text{row}_i| - \mu \\ \mathbf{y}(j) = \sum_{i, (i,j) \in \mathcal{K}} \mathbf{T}(i,j) / |\text{col}_j| - \mu \end{cases} \quad (6)$$

where $|\text{row}_i|$ is the number of the observed elements in the i^{th} row of \mathbf{T} , and $|\text{col}_j|$ is the number of the observed elements in the j^{th} column of \mathbf{T} .

4.3 Discussions and Generalizations

We further present some discussions and generalizations of our optimization formulation.

First, it is worth pointing out that our formulation in Eq. (1) differs from the standard matrix factorization (e.g., SVD) as in the objective function, we try to minimize the square loss *only* on those observed trust pairs. This is because the majority of trust pairs are missing from the input trust matrix \mathbf{T} . In this sense, our problem setting is conceptually similar to the standard collaborative filtering, as in both cases, we aim to fill in missing values in a partially observed matrix (trustor-trustee matrix vs. user-item matrix). Indeed, if we fix the coefficients $\alpha_1 = \alpha_2 = \alpha_3 = 1$ in Eq. (3), it is reduced to the collaborative filtering algorithm in [12]. Our formulation in Eq. (3) is more general as it also allows to learn the optimal coefficients from the input trust matrix \mathbf{T} . Our experimental evaluations show that such subtle treatment is crucial and it leads to further performance improvement over these existing techniques.

Second, although our MaTrust is a subjective trust inference metric where different trustors may form different opinions on the same trustee [20], as a side product, the proposed MaTrust can also be used to infer an objective, unique trustworthiness score for each trustee. For example, this objective trustworthiness score can be computed based on the trustee matrix \mathbf{G} . We will compare this feature of MaTrust with a well studied objective trust inference metric EigenTrust [11] in the experimental evaluation section.

Finally, we would like to point out that our formulation is flexible and can be generalized to other settings. For instance, our current formulation adopts the square loss function in the objective function. In other words, we implicitly assume that the residuals of the pair-wise trustworthiness scores follow a Gaussian distribution, and in our experimental evaluations, we found it works well. Nonetheless, our upcoming proposed MaTrust algorithm can be generalized to *any* Bregman divergence in the objective function. Also, we can naturally incorporate some additional constraints (e.g., non-negativity, sparseness, etc) in the trustor and trustee matrices. After we infer all the parameters (e.g., the coefficients for the bias, and the trustor and trustee matrices, etc), we use a linear combination (i.e., inner product) of the trustor stereotype (i.e., $\mathbf{F}(u, :)$) and trustee stereotype (i.e., $\mathbf{G}(v, :)$) to compute the trustworthiness score $\hat{\mathbf{T}}(u, v)$. We can also generalize this linear form to other non-linear combinations, such as the logistic function. For the sake of clarity, we skip the details of such generalizations in the paper.

5. THE PROPOSED MATRUST ALGORITHM

In this section, we present the proposed algorithm to solve the MaTrust trust inference problem (i.e., Eq. (4)), followed by some effectiveness and efficiency analysis.

Algorithm 1 updateMatrix(\mathbf{P}, r). (See the appendix for the details)

Input: The $n \times n$ matrix \mathbf{P} , and the latent factor size r

Output: The $n \times r$ trustor matrix \mathbf{F}_0 , and the $n \times r$ trustee matrix \mathbf{G}_0

1: $[\mathbf{F}_0, \mathbf{G}_0] = \text{alternatingFactorization}(\mathbf{P}, r);$
2: **return** $[\mathbf{F}_0, \mathbf{G}_0];$

5.1 The MaTrust Algorithm

Unfortunately, the optimization problem in Eq. (4) is not jointly convex wrt the coefficients (α_1, α_2 , and α_3) and the trustor/trustee matrices (\mathbf{F}_0 and \mathbf{G}_0) due to the coupling between them. Therefore, instead of seeking for a global optimal solution, we try to find a local minima by alternatively updating the coefficients and the trustor/trustee matrices while fixing the other. The alternating procedure will lead to a local optima when the convergence criteria are met, i.e., either the L_2 norm between successive estimates of both \mathbf{F} and \mathbf{G} (which are equivalent to $\alpha_1, \alpha_2, \alpha_3, \mathbf{F}_0$, and \mathbf{G}_0) is below our threshold ξ_1 or the maximum iteration step m_1 is reached.

5.1.1 Sub-routine 1: updating the trustor/trustee matrices

First, let us consider how to update the trustor/trustee matrices (\mathbf{F}_0 and \mathbf{G}_0) when we fix the coefficients (α_1, α_2 , and α_3). For clarity, we define an $n \times n$ matrix \mathbf{P} as follows:

$$\mathbf{P}(i, j) = \begin{cases} \mathbf{T}(i, j) - (\alpha_1 \mu + \alpha_2 \mathbf{x}(i) + \alpha_3 \mathbf{y}(j)) & \text{if } (i, j) \in \mathcal{K} \\ '?' & \text{otherwise} \end{cases} \quad (7)$$

where α_1, α_2 , and α_3 are some fixed constants.

Based on the above definition, Eq. (4) can be simplified (by ignoring some constant terms) as:

$$\min_{\mathbf{F}_0, \mathbf{G}_0} \sum_{(i,j) \in \mathcal{K}} (\mathbf{P}(i, j) - \mathbf{F}_0(i, :) \mathbf{G}_0(j, :))'^2 + \lambda \|\mathbf{F}_0\|_{fro}^2 + \lambda \|\mathbf{G}_0\|_{fro}^2 \quad (8)$$

Therefore, updating the trustor/trustee matrices when we fix the coefficients unchanged becomes a standard matrix factorization problem for missing values. Many existing algorithms (e.g., [12, 19, 2]) can be plugged in to solve Eq.(8). In our experiment, we found the so-called alternating strategy, where we recursively update one of the two trustee/trustor matrices while keeping the other matrix fixed, works best and thus recommend it in practice. A brief skeleton of the algorithm is shown in Algorithm 1, and the detailed algorithms are presented in the appendix for completeness.

5.1.2 Sub-routine 2: updating the coefficients

Here, we consider how to update the coefficients (α_1, α_2 , and α_3) when we fix the trustor/trustee matrices.

If we fix the trustor and trustee matrices (\mathbf{F}_0 and \mathbf{G}_0) and let:

$$\mathbf{P}(i, j) = \begin{cases} \mathbf{T}(i, j) - \mathbf{F}_0(i, :) \mathbf{G}_0(j, :)' & \text{if } (i, j) \in \mathcal{K} \\ '?' & \text{otherwise} \end{cases} \quad (9)$$

Eq. (4) can then be simplified (by dropping constant terms) as:

$$\min_{\alpha_1, \alpha_2, \alpha_3} \sum_{(i,j) \in \mathcal{K}} (\mathbf{P}(i, j) - (\alpha_1 \mu + \alpha_2 \mathbf{x}(i) + \alpha_3 \mathbf{y}(j)))^2 + n\lambda \sum_{i=1}^3 \alpha_i^2 \quad (10)$$

To simplify the description, let us introduce another scalar k to index each pair (i, j) in the observed trustor-trustee pairs \mathcal{K} , that is,

Algorithm 2 MaTrust($\mathbf{T}, \mathcal{K}, r, u, v$).

Input: The $n \times n$ partially observed trust matrix \mathbf{T} , the set of observed trustor-trustee pairs \mathcal{K} , the latent factor size r , trustor u , and trustee v

Output: The estimated trustworthiness score $\hat{\mathbf{T}}(u, v)$

```
1:  $[\mu, \mathbf{x}, \mathbf{y}] \leftarrow \text{computeBias}(\mathbf{T})$ ;  
2: initialize  $\alpha_1 = \alpha_2 = \alpha_3 = 1$ ;  
3: while not convergent do  
4:   for each  $(i, j) \in \mathcal{K}$  do  
5:      $\mathbf{P}(i, j) \leftarrow \mathbf{T}(i, j) - (\alpha_1 \mu + \alpha_2 \mathbf{x}(i) + \alpha_3 \mathbf{y}(j))$ ;  
6:   end for  
7:    $[\mathbf{F}_0, \mathbf{G}_0] = \text{updateMatrix}(\mathbf{P}, r)$ ;  
8:   for each  $(i, j) \in \mathcal{K}$  do  
9:      $\mathbf{P}(i, j) \leftarrow \mathbf{T}(i, j) - \mathbf{F}_0(i, :) \mathbf{G}_0(j, :)'$ ;  
10:  end for  
11:   $[\alpha_1, \alpha_2, \alpha_3]' = \text{updateCoefficient}(\mathbf{P}, \mu, \mathbf{x}, \mathbf{y})$ ;  
12: end while  
13: return  $\hat{\mathbf{T}}(u, v) \leftarrow \mathbf{F}_0(u, :) \mathbf{G}_0(v, :)' + \alpha_1 \mu + \alpha_2 \mathbf{x}(u) + \alpha_3 \mathbf{y}(v)$ ;
```

$(i, j) \in \mathcal{K} \rightarrow k = \{1, 2, \dots, |\mathcal{K}|\}$. Let \mathbf{b} denote a vector of length $|\mathcal{K}|$ with $\mathbf{b}(k) = \mathbf{P}(i, j)$. We also define a $|\mathcal{K}| \times 3$ matrix \mathbf{A} as: $\mathbf{A}(k, 1) = \mu$, $\mathbf{A}(k, 2) = \mathbf{x}(i)$, $\mathbf{A}(k, 3) = \mathbf{y}(j)$ ($k = 1, 2, \dots, |\mathcal{K}|$); and a 3×1 vector $\alpha = [\alpha_1, \alpha_2, \alpha_3]'$. Then, Eq. (10) can be formulated as the ridge regression problem wrt the vector α :

$$\min_{\alpha} \|\mathbf{b} - \mathbf{A}\alpha\|_2^2 + n\lambda \|\alpha\|^2 \quad (11)$$

In practice, we shrink the regularization parameter in the above equation from $n\lambda$ to λ to strengthen the importance of bias. Therefore, we can update the coefficients as:

$$\alpha = [\alpha_1, \alpha_2, \alpha_3]' = (\mathbf{A}'\mathbf{A} + \lambda \mathbf{I}_{3 \times 3})^{-1} \mathbf{A}'\mathbf{b} \quad (12)$$

5.1.3 Putting everything together: MaTrust

Putting everything together, we propose Algorithm 2 for our MaTrust trust inference problem. The algorithm first uses Eq. (6) to compute the global bias, trustor bias, and trustee bias (Step 1), and initializes the coefficients (Step 2). Then the algorithm begins the alternating procedure (Step 3-12). First, it fixes α_1 , α_2 , and α_3 , and applies Eq. (7) to incorporate bias. After that, the algorithm invokes Algorithm 1 to update the trustor matrix \mathbf{F}_0 and trustee matrix \mathbf{G}_0 . Next, the algorithm fixes \mathbf{F}_0 and \mathbf{G}_0 , and uses ridge regression in Eq. (12) to update α_1 , α_2 , and α_3 . The alternating procedure ends when the stopping criteria of Eq. (4) are met. Finally, the algorithm outputs the estimated trustworthiness from the given trustor u to the trustee v using Eq. (5) (Step 13).

It is worth pointing out that Step 1-12 in the algorithm can be pre-computed and their results (\mathbf{F}_0 , \mathbf{G}_0 , α_1 , α_2 , and α_3) can be stored in the off-line/pre-computational stage. When an on-line trust inference request arrives, MaTrust only needs to apply Step 13 to return the inference result, which only requires a constant time.

5.2 Algorithm Analysis

Here, we briefly analyze the effectiveness and efficiency of our MaTrust algorithm and provide detailed proofs in the appendix.

The effectiveness of the proposed MaTrust algorithm can be summarized in Lemma 1, which says that overall, it finds a local minima solution. Given that the original optimization problem in Eq. (4) is not jointly convex wrt the coefficients (α_1 , α_2 , and α_3) and the trustor/trustee matrices (\mathbf{F}_0 and \mathbf{G}_0), such a local minima is acceptable in practice.

LEMMA 1. Effectiveness of MaTrust. *Algorithm 2 finds a local minima for the optimization problem in Eq. (4).*

PROOF. See the Appendix □

The time complexity of the proposed MaTrust is summarized in Lemma 2, which says that MaTrust scales *linearly* wrt the number of users and the number of the observed trustor-trustee pairs.

LEMMA 2. Time Complexity of MaTrust. *Algorithm 2 requires $O(nr^2m_1m_2 + |\mathcal{K}|r^2m_1m_2)$ time, where m_1 and m_2 are the maximum iteration numbers in Algorithm 2 and Algorithm 1, respectively.*

PROOF. See the Appendix □

The space complexity of MaTrust is summarized in Lemma 3, which says that MaTrust requires *linear* space wrt the number of users and the number of the observed trustor-trustee pairs.

LEMMA 3. Space Complexity of MaTrust. *Algorithm 2 requires $O(|\mathcal{K}| + nr + r^2)$ space.*

PROOF. See the Appendix □

Notice that for both time complexity and space complexity, we have a *polynomial* term wrt the number of the latent factors r . In practice, this parameter is small compared with the number of the users (n) or the number of the observed trustor-trustee pairs ($|\mathcal{K}|$). For example, in our experiments, we did not observe significant performance improvement when the number of latent factors is larger than 10 (See the next section for the detailed evaluations).

6. EXPERIMENTAL EVALUATION

In this section, we present experimental evaluations, after we introduce the data sets. All the experiments are designed to answer the following questions:

- *Effectiveness*: How accurate is the proposed MaTrust for trust inference? How robust is the inference result wrt the different parameters in MaTrust?
- *Efficiency*: How fast is the proposed MaTrust? How does it scale?

6.1 Data Sets Description

Many existing trust inference models design specific simulation studies to verify the underlying assumptions of the corresponding inference models. In contrast, we focus on two widely used real, benchmark data sets in order to compare the performance of different trust inference models.

The first data set is *advogato*¹. It is a trust-based social network for open source developers. To allow users to certify each other, the network provides 4 levels of trust assertions, i.e., ‘*Observer*’, ‘*Apprentice*’, ‘*Journeyer*’, and ‘*Master*’. These assertions can be mapped into real numbers which represent the degree of trust. To be specific, we map ‘*Observer*’, ‘*Apprentice*’, ‘*Journeyer*’, and ‘*Master*’ to 0.1, 0.4, 0.7, and 0.9, respectively (a higher value means more trustworthiness).

¹http://www.trustlet.org/wiki/Advogato_dataset.

Table 2: High level statistics of advogato and PGP data sets.

Data set	Nodes	Edges	Avg. degree	Avg. clustering [33]	Avg. diameter [15]	Date
advogato-1	279	2,109	15.1	0.45	4.62	2000-02-05
advogato-2	1,261	12,176	19.3	0.36	4.71	2000-07-18
advogato-3	2,443	22,486	18.4	0.31	4.67	2001-03-06
advogato-4	3,279	32,743	20.0	0.33	4.74	2002-01-14
advogato-5	4,158	41,308	19.9	0.33	4.83	2003-03-04
advogato-6	5,428	51,493	19.0	0.31	4.82	2011-06-23
PGP	38,546	317,979	16.5	0.45	7.70	2008-06-05

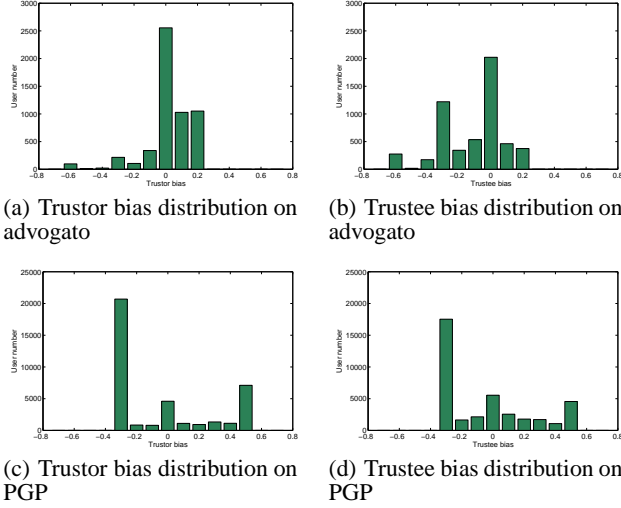


Figure 2: The distributions of trustor bias and trustee bias.

The second data set is *PGP* (short for Pretty Good Privacy) [8]. PGP adopts the concept of ‘web of trust’ to establish a decentralized model for data encryption and decryption. Similar to *advogato*, the web of trust in PGP data set contains 4 levels of trust as well. In our experiments, we also map them to 0.1, 0.4, 0.7, and 0.9, respectively.

Table 2 summarizes the basic statistics of the two resulting partially observed trust matrices \mathbf{T} . Notice that for the *advogato* data set, it contains six different snapshots, i.e., *advogato-1*, *advogato-2*, ..., *advogato-6*, etc. We use the largest snapshot (i.e., *advogato-6*) in the following unless otherwise specified.

Fig. 2 presents the distributions of trustor bias and trustee bias. As we can see, many users in *advogato* perform averagely on trusting others and being trusted by others. On the other hand, a considerable part of PGP users are cautiously trusted by others, and even more users tend to rate others strictly. The global bias is 0.6679 and 0.3842 for *advogato* and PGP, respectively. This also confirms that the security-related PGP network is a more conservative environment than the developer-based *advogato* network.

6.2 Effectiveness Results

We use both *advogato* (i.e., *advogato-6*) and *PGP* for effectiveness evaluations. For both data sets, we hide a randomly selected sample of 500 observed trustor-trustee pairs as the test set, and apply the proposed MaTrust as well as other compared methods on the remaining data set to infer the trustworthiness scores for those hidden pairs. To evaluate and compare the accuracy, we report both

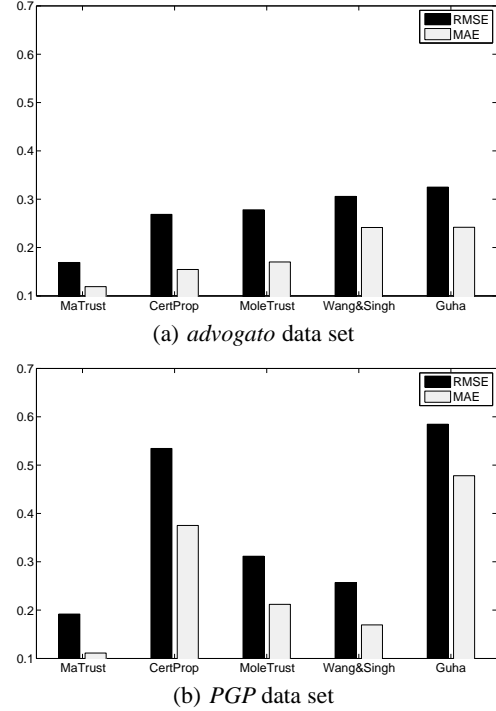


Figure 3: Comparisons with subjective trust inference models. The proposed MaTrust significantly outperforms all the other models wrt both RMSE and MAE on both data sets.

the root mean squared error (RMSE) and the mean absolute error (MAE) between the estimated and the true trustworthiness scores. Both RMSE and MAE are measured on the 500 hidden pairs in the test set. We set $\lambda = 1.0$, $r = 10$, $m_1 = 10$, $m_2 = 100$, and $\xi_1 = \xi_2 = 10^{-6}$ in our experiments unless otherwise specified.

(A) *Comparisons with Existing Subjective Trust Inference Methods.* We first compare the effectiveness of MaTrust with several benchmark trust propagation models, including *CertProp* [8], *MoleTrust* [20], *Wang&Singh* [31, 32], and *Guha* [7]. For all these subjective methods, the goal is to infer a pair-wise trustworthiness score (i.e., to what extent the user u trusts another user v).

The result is shown in Fig. 3. We can see that the proposed MaTrust significantly outperforms all the other trust inference models wrt both RMSE and MAE on both data sets. For example, on *advogato* data set, our MaTrust improves the best existing method (*CertProp*) by 37.1% in RMSE and by 23.0% in MAE. As for *PGP* data set, the proposed MaTrust improves the best existing method (*Wang&Singh*) by 25.3% in RMSE and by 34.3% in MAE. The

Table 3: Comparisons with *EigenTrust*. MaTrust is better than EigenTrust wrt both RMSE and MAE on both data sets.

RMSE/MAE	advogato	PGP
EigenTrust	0.700 / 0.653	0.519 / 0.371
MaTrust	0.290 / 0.203	0.349 / 0.280

Table 4: The importance of trust bias. Trust bias significantly improves trust inference accuracy.

RMSE/MAE	advogato	PGP
MaTrust without trust bias	0.228 / 0.164	0.244 / 0.135
MaTrust	0.169 / 0.119	0.192 / 0.111

Table 5: Comparisons with *SVD* and *KBV* [12]. MaTrust outperforms both of them.

RMSE/MAE	advogato	PGP
SVD	0.629 / 0.579	0.447 / 0.306
KBV	0.179 / 0.125	0.217 / 0.133
MaTrust	0.169 / 0.119	0.192 / 0.111

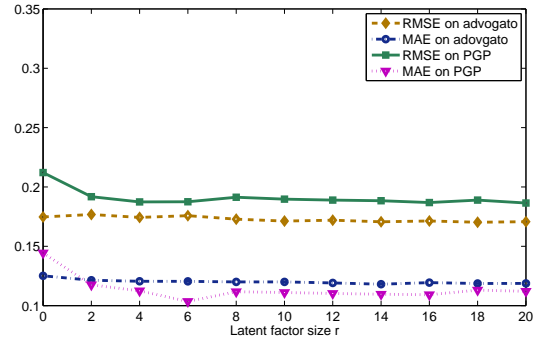
results suggest that multi-aspect of trust indeed plays a very important role in the inference process.

(B) *Comparisons with Existing Objective Trust Inference Methods.* Although our MaTrust is a subjective trust inference metric, as a side product, it can also be used to infer an objective trustworthiness score for each trustee. To this end, we set $r = 1$ in MaTrust algorithm, and aggregate the resulting trustee matrix/vector \mathbf{G}_0 with the bias (the global bias μ and the trustee bias \mathbf{y}). We compare the result with a widely-cited objective trust inference model *EigenTrust* [11] in Table 3. As we can see, MaTrust outperforms EigenTrust in terms of both RMSE and MAE on both data sets. For example, on *advogato* data set, MaTrust is 58.6% and 68.9% better than EigenTrust wrt RMSE and MAE, respectively.

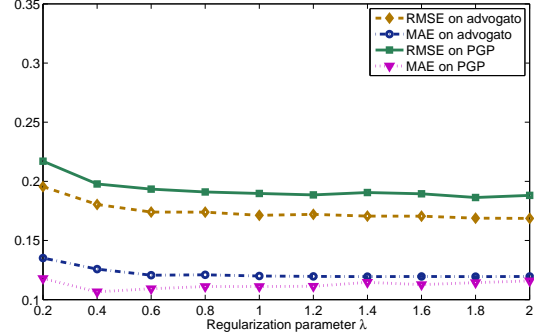
(C) *Trust Bias Evaluations.* We next show the importance of trust bias by comparing MaTrust with the results when trust bias is not incorporated. The result is shown in Table 4. As we can see, MaTrust performs much better when trust bias is incorporated. For example, on *PGP* data set, trust bias helps MaTrust to obtain 21.3% and 17.8% improvements in RMSE and MAE, respectively. This result confirms that trust bias also plays an important role in trust inference.

(D) *Comparisons with Existing Matrix Factorization Methods.* We also compare MaTrust with two existing matrix factorization methods: *SVD* and the collaborative filtering algorithm [12] for recommender systems (referred to as *KBV*).

The result is shown in Table 5. As we can see from the table, MaTrust again outperforms both *SVD* and *KBV* on both data sets. *SVD* performs poorly as it treats all the unobserved trustor-trustee pairs as zero elements in the trust matrix \mathbf{T} . MaTrust also outperforms *KBV*. For example, MaTrust improves *KBV* by 11.5% in RMSE and by 16.5% in MAE on *PGP* data set. As mentioned before, *KBV* can be viewed as a special case of the proposed MaTrust if we fix all the coefficients as 1. This result confirms that by simultaneously learning the bias coefficients from the input trust matrix \mathbf{T} (i.e., the relative weights for different types of bias), MaTrust leads to further performance improvement.



(a) RMSE and MAE of MaTrust wrt r . We fix $r = 10$.



(b) RMSE and MAE of MaTrust wrt λ . We fix $\lambda = 1.0$.

Figure 4: The sensitivity evaluations. MaTrust is robust wrt both parameters.

(E) *Sensitivity Evaluations.* Finally, we conduct a parametric study for MaTrust. The first parameter is the latent factor size r . We can observe from Fig. 4(a) that, in general, both RMSE and MAE stay stable wrt r with a slight decreasing trend. For example, compared with the results of $r = 2$, the RMSE and MAE decrease by 3.1% and 4.3% on average if we increase $r = 20$. The second parameter in MaTrust is the regularization coefficient λ . As we can see from Fig. 4(b), both RMSE and MAE decrease when λ increases up to 0.8; and they stay stable after $\lambda > 0.8$. Based on these results, we conclude that MaTrust is robust wrt both parameters. For all the other results we report in the paper, we simply fix $r = 10$ and $\lambda = 1.0$.

6.3 Efficiency Results

For efficiency experiments, we report the average wall-clock time. All the experiments were run on a machine with two 2.4GHz Intel Cores and 4GB memory.

(A) *Speed Comparison.* We first compare the on-line response of MaTrust with *CertProp*, *MoleTrust*, *Wang&Singh*, and *Guha*. Again, we use the *advogato-6* snapshot and *PGP* in this experiment, and the result is shown in Fig. 5. Notice that the y-axis is in the logarithmic scale.

We can see from the figure that the proposed MaTrust is much faster than all the alternative methods on both data sets. For example, MaTrust is 2,000,000 - 3,500,000x faster than *MoleTrust*. This is because once we have inferred the trustor/trustee matrices as well as the coefficients for the bias (Step 1-12 in Algorithm 2), it only takes *constant* time for MaTrust to output the trustworthiness score (Step 13 in Algorithm 2). Among all the alternative methods, *Guha*

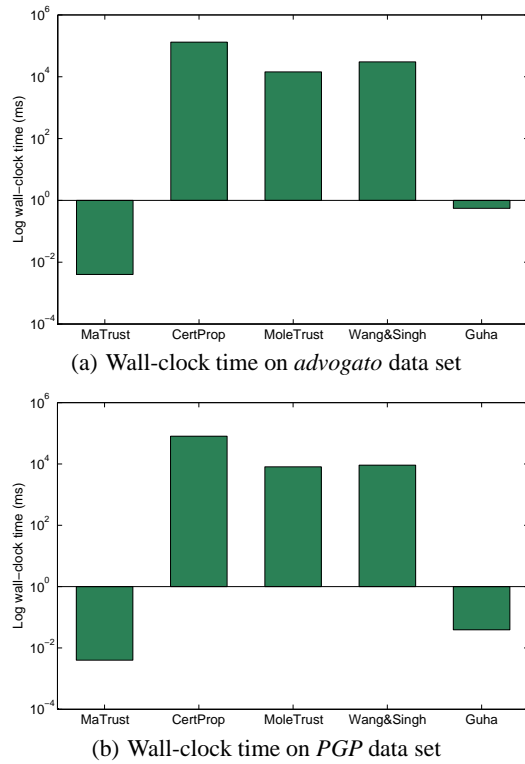


Figure 5: Speed comparison. MaTrust is much faster than all the other methods.

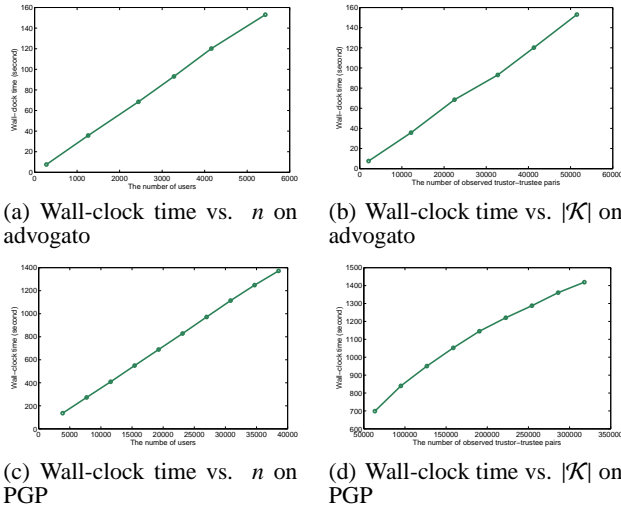


Figure 6: Scalability of the proposed MaTrust. MaTrust scales linearly wrt the data size (n and $|\mathcal{K}|$).

is the most efficient. This is because its main workload can also be completed in advance. However, the pre-computation of Guha needs additional $O(n^2)$ space as the model fills nearly all the missing elements in the trust matrix, making it unsuitable for large data sets. In contrast, MaTrust only requires $O(|\mathcal{K}| + nr + r^2)$ space, which is usually much smaller than n^2 .

(B) *Scalability.* Finally, we present the scalability result of MaTrust by reporting the wall-clock time of the pre-computational

stage (i.e., Step 1-12 in Algorithm 2). For *advogato* data set, we directly report the results on all the six snapshots (i.e., *advogato-1*, ..., *advogato-6*). For *PGP*, we use its subsets to study the scalability. The result is shown in Fig. 6, which is consistent with the complexity analysis in Section 5.2. As we can see from the figure, MaTrust scales linearly wrt to both n and $|\mathcal{K}|$, indicating that it is suitable for large-scale applications.

7. CONCLUSION

In this paper, we have proposed an effective multi-aspect trust inference model (MaTrust). The key idea of MaTrust is to characterize several aspects/factors for each trustor and trustee based on the existing trust relationships. The proposed MaTrust can naturally incorporate the prior knowledge such as trust bias by expressing it as specified factors. In addition, MaTrust scales linearly wrt the input data size (e.g., the number of users, the number of observed trustor-trustee pairs, etc). Our experimental evaluations on real data sets show that trust bias can truly improve the inference accuracy, and that MaTrust significantly outperforms existing benchmark trust inference models in both effectiveness and efficiency. Future work includes investigating the capability of MaTrust to address the distrust as well as the trust dynamics.

8. REFERENCES

- [1] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD*, pages 95–104. ACM, 2007.
- [2] A. Buchanan and A. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *CVPR*, volume 2, pages 316–322, 2005.
- [3] C. Burnett, T. Norman, and K. Sycara. Bootstrapping trust evaluations through stereotypes. In *AAMAS*, pages 241–248, 2010.
- [4] K. Chiang, N. Natarajan, A. Tewari, and I. Dhillon. Exploiting longer cycles for link prediction in signed networks. In *CIKM*, pages 1157–1162, 2011.
- [5] D. Gefen. Reflections on the dimensions of trust and trustworthiness among online consumers. *ACM SIGMIS Database*, 33(3):38–53, 2002.
- [6] J. Golbeck. Trust and nuanced profile similarity in online social networks. *ACM Transactions on the Web*, 3(4):12, 2009.
- [7] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW*, pages 403–412. ACM, 2004.
- [8] C.-W. Hang, Y. Wang, and M. P. Singh. Operators for propagating trust and their evaluation in social networks. In *AAMAS*, pages 1025–1032, 2009.
- [9] C. Hsieh, K. Chiang, and I. Dhillon. Low rank modeling of signed networks. In *KDD*, 2012.
- [10] A. Jøsang and R. Ismail. The Beta reputation system. In *Proc. of the 15th Bled Electronic Commerce Conference*, volume 160, Bled, Slovenia, June 2002.
- [11] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in p2p networks. In *WWW*, pages 640–651. ACM, 2003.
- [12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [13] U. Kuter and J. Golbeck. Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models. In *AAAI*, pages 1377–1382, 2007.

- [14] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *WWW*, pages 641–650. ACM, 2010.
- [15] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD*, pages 177–187. ACM, 2005.
- [16] G. Liu, Y. Wang, and M. Orgun. Optimal social trust path selection in complex social networks. In *AAAI*, pages 1391–1398, 2010.
- [17] G. Liu, Y. Wang, and M. Orgun. Trust transitivity in complex social networks. In *AAAI*, pages 1222–1229, 2011.
- [18] X. Liu, A. Datta, K. Rzadca, and E. Lim. Stereotrust: a group based personalized trust model. In *CIKM*, pages 7–16. ACM, 2009.
- [19] H. Ma, M. Lyu, and I. King. Learning to recommend with trust and distrust relationships. In *RecSys*, pages 189–196. ACM, 2009.
- [20] P. Massa and P. Avesani. Controversial users demand local trust metrics: An experimental study on opinions. com community. In *AAAI*, pages 121–126, 2005.
- [21] A. Mishra and A. Bhattacharya. Finding the bias and prestige of nodes in networks based on trust scores. In *WWW*, pages 567–576. ACM, 2011.
- [22] V. Nguyen, E. Lim, J. Jiang, and A. Sun. To trust or not to trust? predicting online trusts using trust antecedent framework. In *ICDM*, pages 896–901. IEEE, 2009.
- [23] K. Nordheimer, T. Schulze, and D. Veit. Trustworthiness in networks: A simulation approach for approximating local trust and distrust values. In *IFIPTM*, volume 321, pages 157–171. Springer-Verlag, 2010.
- [24] R. Paul and Z. Richard. Trust among strangers in Internet transactions: Empirical analysis of eBay’s reputation system. In *The Economics of the Internet and E-Commerce*, volume 11 of *Advances in Applied Microeconomics: A Research Annual*, pages 127–157. Elsevier, 2002.
- [25] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the Semantic Web. In *ISWC*, pages 351–368. Springer, 2003.
- [26] J. Sabater and C. Sierra. Reputation and social network analysis in multi-agent systems. In *AAMAS*, pages 475–482. ACM, 2002.
- [27] D. Sirdeshmukh, J. Singh, and B. Sabol. Consumer trust, value, and loyalty in relational exchanges. *The Journal of Marketing*, pages 15–37, 2002.
- [28] J. Tang, H. Gao, and H. Liu. mTrust: discerning multi-faceted trust in a connected world. In *WSDM*, pages 93–102. ACM, 2012.
- [29] A. Tversky and D. Kahneman. Judgment under uncertainty: Heuristics and biases. *science*, 185(4157):1124–1131, 1974.
- [30] G. Wang and J. Wu. Multi-dimensional evidence-based trust management with multi-trusted paths. *Future Generation Computer Systems*, 27(5):529–538, 2011.
- [31] Y. Wang and M. P. Singh. Trust representation and aggregation in a distributed agent system. In *AAAI*, pages 1425–1430, 2006.
- [32] Y. Wang and M. P. Singh. Formal trust model for multiagent systems. In *IJCAI*, pages 1551–1556, 2007.
- [33] D. Watts and S. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684):440–442, 1998.
- [34] L. Xiong and L. Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE*

Transactions on Knowledge and Data Engineering, 16(7):843–857, 2004.

- [35] Y. Yao, H. Tong, F. Xu, and J. Lu. Subgraph extraction for trust inference in social networks. In *ASONAM (to appear)*, 2012.
- [36] C. Ziegler and G. Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4):337–358, 2005.

APPENDIX

A. DETAILED ALGORITHM 1

Here, we present the complete algorithm to update the trustor/trustee matrices when the bias coefficients are fixed (i.e., Algorithm 1 for Eq. (8)). As mentioned above, we apply the alternating strategy by alternatively fixing one of the two matrices and optimizing the other. For simplicity, let us consider how to update \mathbf{F}_0 when \mathbf{G}_0 is fixed. Updating \mathbf{G}_0 when \mathbf{F}_0 is fixed can be done in a similar way. By fixing \mathbf{G}_0 , Eq. (8) can be further simplified as follows:

$$\min_{\mathbf{F}_0} \sum_{(i,j) \in \mathcal{K}} (\mathbf{P}(i, j) - \mathbf{F}_0(i, :) \mathbf{G}_0(j, :)^T)^2 + \lambda \|\mathbf{F}_0\|_{fro}^2 \quad (13)$$

In fact, the above optimization problem in Eq. (13) now becomes convex wrt \mathbf{F}_0 . It can be further decoupled into many independent sub-problems, each of which only involves a single row in \mathbf{F}_0 :

$$\min_{\mathbf{F}_0(i,:)} \sum_{j, (i,j) \in \mathcal{K}} (\mathbf{P}(i, j) - \mathbf{F}_0(i, :) \mathbf{G}_0(j, :)^T)^2 + \lambda \|\mathbf{F}_0(i, :)\|^2 \quad (14)$$

The optimization problem in Eq. (14) can now be solved by the standard ridge regression wrt the corresponding row $\mathbf{F}_0(i, :)$.

Algorithm 3 `alternateUpdate(P, F0, G0)`.

Input: The $n \times n$ matrix \mathbf{P} , the $n \times r$ matrix \mathbf{F}_0 , and the fixed $n \times r$ matrix \mathbf{G}_0

Output: The updated matrix \mathbf{F}_1 of \mathbf{F}_0

```

1:  $\mathbf{F}_1 \leftarrow \mathbf{F}_0$ ;
2: for  $i = 1 : n$  do
3:    $\mathbf{a} \leftarrow$  the vector of column indices of existing elements in
      $\mathbf{P}(i, j)$  ( $j = 1, 2, \dots, n$ );
4:   column vector  $\mathbf{d} \leftarrow \mathbf{0}_{|a| \times 1}$ ;
5:   matrix  $\mathbf{G}_1 \leftarrow \mathbf{0}_{|a| \times r}$ ;
6:   for  $j = 1 : |a|$  do
7:      $\mathbf{d}(j) \leftarrow \mathbf{P}(i, \mathbf{a}(j))$ ;
8:      $\mathbf{G}_1(j, :) \leftarrow \mathbf{G}_0(\mathbf{a}(j), :)$ ;
9:   end for
10:   $\mathbf{F}_1(i, :) \leftarrow (\mathbf{G}_1' \mathbf{G}_1 + \lambda \cdot \mathbf{I}_{r \times r})^{-1} \mathbf{G}_1' \mathbf{d}$ ;
11: end for
12: return  $\mathbf{F}_1$ ;

```

Algorithm 3 presents the overall solution for updating the trustor matrix \mathbf{F}_0 . Based on Algorithm 3, we present Algorithm 4 to alternatively update the trustor and trustee matrices \mathbf{F}_0 and \mathbf{G}_0 . The algorithm first generates two $n \times r$ matrices for \mathbf{F}_0 and \mathbf{G}_0 where each element is initialized as $1/r$. At each iteration, the algorithm then alternatively calls Algorithm 3 to update the two matrices. The iteration ends when the stopping criteria are met, i.e., either the L_2 norm between successive estimates of both \mathbf{F}_0 and \mathbf{G}_0 is below our threshold ξ_2 or the maximum iteration step m_2 is reached.

Algorithm 4 updateMatrix(\mathbf{P} , r).

Input: The $n \times n$ matrix \mathbf{P} , and the latent factor size r

Output: The $n \times r$ trustor matrix \mathbf{F}_0 , and the $n \times r$ trustee matrix \mathbf{G}_0

```

1: generate the  $n \times r$  matrices  $\mathbf{F}_0$  and  $\mathbf{G}_0$  randomly;
2: while not convergent do
3:    $\mathbf{F}_0 \leftarrow \text{alternateUpdate}(\mathbf{P}, \mathbf{F}_0, \mathbf{G}_0)$ ;
4:    $\mathbf{G}_0 \leftarrow \text{alternateUpdate}(\mathbf{P}', \mathbf{G}_0, \mathbf{F}_0)$ ;
5: end while
6: return  $[\mathbf{F}_0, \mathbf{G}_0]$ ;

```

B. PROOFS FOR LEMMAS

Next, we present the proofs for the lemmas in Section 5.2.

Proof Sketch for Lemma 1: (P1) First, Eq. (14) is convex and therefore Step 10 in Algorithm 3 finds the global optima for updating a single row in the matrix \mathbf{F}_0 . Notice that the optimization problem in Eq. (14) is equivalent to that in Eq. (13), and thus we have proved that Algorithm 3 finds the global optimal solution for the optimization problem in Eq. (13).

(P2) Next, based on (P1) and the alternating procedure in Algorithm 4, we have that Algorithm 4 finds a local minima for the optimization problem in Eq. (8).

(P3) Finally, based on (P2) and the alternating procedure in Algorithm 2, Lemma 1 holds. \square

Proof of Lemma 2: (P1) In Algorithm 3, the time cost for Step 1 is $O(nr)$. Let a_i denote the number of elements in \mathbf{a} of the i^{th} iteration. The time cost for Step 3-5 is then $O(a_i r)$ since we store \mathbf{P} in sparse format. We need another $O(a_i r)$ time in the inner iteration (Step 6-9). The time cost of Step 10 is $O(a_i r^2 + r^2 + r^3 + a_i r^2 + a_i r + r) = O(r^3 + a_i r^2)$. Therefore, the total time cost for the algorithm is $O(nr) + O(\sum_i (r^3 + a_i r^2)) = O(nr^3 + |\mathcal{K}|r^2)$ where $\sum_i a_i = |\mathcal{K}|$.

(P2) In Algorithm 4, the time cost for Step 1 is $O(nr)$. As indicated by (P1), we need $O(nr^3 + |\mathcal{K}|r^2)$ time for both Step 3 and Step 4. The total time cost is $O(nr^3 m_2 + |\mathcal{K}|r^2 m_2)$.

(P3) In Algorithm 2, the time cost for Step 1 is $O(|\mathcal{K}|)$ as we store \mathbf{T} in sparse format. Step 2 needs $O(1)$ time. We need $O(|\mathcal{K}|)$ time for Step 4-6. As indicated by (P2), we need $O(nr^3 m_2 + |\mathcal{K}|r^2 m_2)$ time for Step 7. We need $O(|\mathcal{K}|r)$ time for Step 8-10. As for updating the coefficients, we need $O(|\mathcal{K}|c^2 + c^3)$ time where c is the number of specified factors, which is 3 in our case. Therefore, the time cost for Step 11 is $O(|\mathcal{K}|)$. The total time cost is $O(nr^3 m_1 m_2 + |\mathcal{K}|r^2 m_1 m_2)$, which completes the proof. \square

Proof of Lemma 3: (P1) In Algorithm 3, we need $O(nr)$ space for Step 1 and $O(1)$ space for Step 2. We need another $O(nr)$ space for Step 3-5. For Step 6-9 we only need $O(1)$ space. We need $O(nr + r^2)$ space for Step 10. Among the different iterations of the algorithm, we can re-use the space from the previous iteration. Finally, the overall space cost is $O(|\mathcal{K}| + nr + r^2)$.

(P2) In Algorithm 4, we need $O(nr)$ space for Step 1. Step 3 and Step 4 need $O(|\mathcal{K}| + nr + r^2)$ space. The space for each iteration can be re-used. The total space cost is $O(|\mathcal{K}| + nr + r^2)$.

(P3) In Algorithm 2, we need $O(|\mathcal{K}|)$ space for the input since we store \mathbf{T} in sparse format. We need $O(n)$ space for Step 1 and $O(1)$

space for the Step 2. We need another $O(|\mathcal{K}|)$ space for Step 4-6. By (P2), Step 7 needs $O(|\mathcal{K}| + nr + r^2)$ space. Step 8-10 can re-use the space from Step 4-6. Step 11 needs $O(|\mathcal{K}|)$ space. For each iteration, the space can be re-used. The total space cost of Algorithm 2 is $O(|\mathcal{K}| + nr + r^2)$, which completes the proof. \square